To appear in the ACM APGV 2007 symposium proceedings

Hybrid Image-/Model-Based Gaze-Contingent Rendering

Hunter Murphy Andrew T. Duchowski *

School of Computing, Clemson University

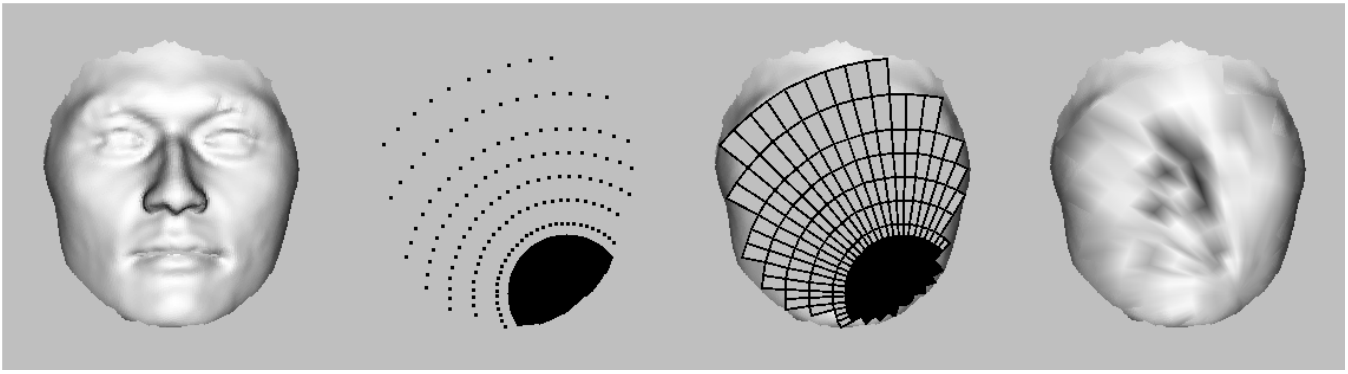

Figure 1: *The original geometry is sampled via ray casting, silhouette edges are identified, and the peripherally degraded image is displayed.*

Abstract

A nonisotropic hybrid image-/model-based gaze-contingent rendering technique utilizing ray casting on a GPU is discussed. Empirical evidence derived from human subject experiments indicates an inverse relationship between a peripherally degraded scene's high-resolution inset size and mean search time, a trend consistent with existing image- and model- based techniques. In addition, the data suggest that maintaining a target's silhouette edges decreases search times when compared to targets with degraded edges. Benefits of the hybrid technique include simplicity of design and parallelizability, both conducive to GPU implementation.

CR Categories: I.3.3 [Computer Graphics]: Picture/Image Generation—Display algorithms; I.3.6 [Computer Graphics]: Methodology and Techniques—Ergonomics;

Keywords: gpu, perceptual rendering, gaze-contingent rendering

1 Introduction

The contributions of this paper are the development and experimental evaluation of a hybrid image-/model-based nonisotropic gaze-contingent rendering technique. This method, based on ray casting, is particularly well suited to implementation on modern programmable GPUs. The evaluation takes the form of a human subject experiment involving a visual search task, the results of which are compared to previous gaze-contingent rendering experiments.

*Email: {hmurphy | andrewd}@cs.clemson.edu

The idea of gaze-contingent rendering is not new, with military applications assuming some of the earliest instances [Duchowski 2007]. Since early forays into gaze-contingent rendering, significant progress has been made in exploring both the extent of peripheral degradation permitted by the Human Visual System (HVS), and the algorithms used to perform that degradation. The field has become dominated by two approaches in particular: image-based rendering and model-based rendering. This paper introduces a third technique which combines the strengths of both image- and model-based rendering in a manner conducive to implementation on modern graphics hardware.

The paper is organized as follows. Section 2 explores the two divergent approaches taken in gaze-contingent rendering. The unification of these disparate techniques is discussed in Section 3. Details regarding the experimental paradigm, data collection, and analysis are presented in Section 4, with the results, discussion, and conclusions appearing in Sections 5, 6, and 7, respectively.

2 Background

Gaze-contingent rendering exploits the human visual system's reduction of peripheral sensitivity to several visual attributes, with the goal of improving rendering performance. The measurement of performance can be ambiguous, but usually falls into two categories: visually or functionally imperceptible degradation.

Visually imperceptible degradation attempts to reduce peripheral information in a manner that cannot be distinguished from a non-degraded version. This stringent approach requires strict adherence to all aspects of the HVS simultaneously. Use of visual imperceptibility as a performance metric is typically applied to evaluation of the HVS itself; e.g., empirical determination of a function describing the imperceptible extent of peripheral chromatic degradation.

Functionally imperceptible degradation takes a more practical approach by interpreting imperceptible degradation to mean that the reduction in peripheral information does not result in reduced task performance. This less rigid stance allows much greater flexibility in manipulating the HVS as it is focused on exploiting, rather than deriving, fundamental HVS knowledge.

2.1 Image-based GCD

An image-based Gaze-Contingent Display (GCD) is particularly well suited to exploring the perceptual limits of the HVS. Implementation typically involves application of a convolution filter to a pre-rendered full resolution image. This simplifies not only the programming aspect, as 2-D filtering is relatively simple, but also the experimental aspect, since starting with an ideal image allows the experimenter to focus on manipulating a single perceptual variable of the HVS.

Sophisticated approaches have been developed for image and video coding [Parkhurst and Niebur 2002; Reingold et al. 2002; Geisler et al. 2006; Duchowski and Çöltekin 2007]. For screen-based rendering the work of Watson et al. [1997] is particularly relevant. The authors studied the effects of LOD peripheral degradation on visual search performance. Both spatial and chrominance detail degradation effects were evaluated in head mounted displays. To sustain acceptable frame rates, two polygons were texture mapped in real-time to generate a high resolution inset within a low resolution display field. The authors suggested that visual spatial and chrominance complexity can be reduced by almost half without degrading performance.

In a similar approach, Reddy [1998] used a view-dependent screen-based LOD technique to evaluate both perceptual effects and system performance gains. The author reported a perceptually modulated LOD system which affords a factor 4.5 improvement in frame rate. It is not entirely clear how the LOD model was constructed, i.e., what was the method of degradation.

More recently, Watson et al. [2004] point out that speed improvement may be limited if only a sub-threshold approach is used to reduce detail.

By filtering out high frequency information in the periphery, image-based GCD can also offer a performance enhancement. The increase in image compressibility gained through peripheral degradation can be exploited to improve bandwidth utilization for image transmission [Bergström 2003].

Despite recent interest, real-time computational benefits of image-based GCD are lacking. Unless used for remote display, there can be no improvement in render time when the first step is to render the scene in full resolution, and the second is to apply a full screen convolution filter. Indeed, outside its uses in psychophysical research and image transmission, image-based GCD offers no practical benefit per se. Model-based GCDs attempt to address the desire for render time efficiency.

2.2 Model-based GCD

A model-based GCD can significantly improve render time by removing mesh geometry which is not perceived by the user. The reduced complexity of geometry to be rendered is either generated from the original mesh through a series of edge collapses or vertex decimations, or is built up from a base mesh until it satisfies the requirements of the HVS. Whether using a top-down or bottom-up approach, model-based GCD typically requires significant processing, either as a preprocessing step or at run-time, but can provide a net speedup in rendering time [Luebke and Erikson 1997].

Simplification of geometric objects to reflect perceptual limits is a widely used technique, dating back to Clark's [1976] description of Level Of Detail (LOD) rendering, wherein an object's screen area coverage is used as a metric to select one of several pre-computed reduced resolution meshes for display. This technique is currently used in applications ranging from animation rendering to virtual reality; however, the approaches used to generate the reduced resolution meshes typically result in isotropic, or uniform, object simplification.

LOD rendering has also been used in model-based GCD. Ohshima et al. [1996] proposed a scheme considering three visual characteristics: central/peripheral vision, kinetic vision, and fusional vision. The LOD algorithm generated isotropically degraded objects at different visual angles. Although the use of a binocular eye tracker was proposed, the system as discussed used only head tracking as a substitute for gaze tracking.

Traditional LOD is clearly beneficial when rendering for a standard display paradigm; the use of isotropic object degradation for GCD is suboptimal. Uniform mesh degradation assumes uniform perceptual discrimination across the entire field of view. In this case, traditional LOD schemes will display an LOD mesh at its full resolution even though the mesh may cover the entire field of view. Since acute resolvability of human vision is limited to the foveal 5°, object resolution need not be uniform within a gaze-contingent context. This is the central tenet of gaze-contingent systems.

An alternative to isotropic LOD, multiresolution mesh modeling techniques suitable for gaze-contingent viewing have been developed [Zorin and Schröder 2000]. Multiresolution modeling allows the possibility of rendering meshes that are non-isotropically degraded by selecting the geometry to display from a hierarchy of meshes based on the angle subtended from the Point Of Regard, or POR. Techniques range from multiresolution representation of arbitrary meshes to the management of LOD through peripheral degradation within an HMD, where gaze position is assumed to coincide with head direction [Lindstrom et al. 1996; MacCracken and Joy 1996; Hoppe 1997; Zorin et al. 1997; Schmalstieg and Schaufler 1997].

Danforth et al. [2000] developed a non-isotropic gaze-contingent multiresolution terrain navigation environment. A surface, represented as a quadrilateral mesh, was divided into fixed-size (number of vertexes) sub-blocks, allowing rendering for variable LOD on a per-sub-block basis. Resolution level was chosen per sub-block, based on viewer distance. The resolution level was not discrete; it was interpolated between the pre-computed discrete levels to avoid "popping" effects. The approach used is reasonably effective; however, it is not clear whether the technique is applicable to arbitrary meshes.

Rather than decimate existing geometry, Murphy and Duchowski [2001] constructed a mesh in real-time from a base mesh. Distance from the POR indicated how much geometry was to be added to the mesh, with additions being made from a hierarchy of intermediate forms.

Unfortunately, model-based GCDs lack the inherent implementational simplicity of image-based GCDs. Reconstruction of mesh geometry, either through decimation of the original mesh or reconstitution of a base mesh, is an issue of local mesh connectivity; however, maintaining the global perceptual requirements makes the utilization of model-based GCD a non-trivial task. Moreover, model-based LOD manipulation may not be applicable to arbitrary meshes and may require constraints on mesh connectivity (e.g., 2-manifold) or mesh reparameterization [Luebke et al. 2000].

In an effort to provide the implementation ease of image-based GCDs, while offering hope of render time efficiencies corresponding to a model-based GCD, a hybrid approach is discussed in the following section. The technique is similar to an early implementation by Levoy and Whitaker [1990], in which the authors ray traced volumetric data in a perceptually adaptive manner. Although rays were not cast in accordance with any particular HVS function, the

number of rays cast increased at the POR and decreased with increasing visual angle according to a Gaussian function. The resulting samples were used to index a 2-D mipmap of images and combined to construct the final image.

3 Hybrid Image- /Model- based GCD

The goal of the hybrid technique is to non-destructively sample scene geometry in a manner consistent with the limits of the HVS. This is achieved using ray casting, with ray distribution conforming to the angular frequency dictated by a Contrast Sensitivity Function (CSF). The ray casting and CSF combination allows non-isotropic (within mesh) degradation without directly manipulating mesh geometry (edge collapse or vertex decimation). An intermediate mesh existing between the eye position and the scene geometry provides both a direction vector for the rays and storage for the resulting ray/primitive intersection data. Further refinement of the intermediate mesh is performed to maintain silhouette edges. The intermediate mesh is ultimately rendered in place of the scene geometry.

3.1 Contrast Sensitivity Function

A contrast sensitivity function attempts to describe the amount of detail visible at increasing angular separation from the point of regard (e.g., see Luebke and Hallen [2001]). Contrast is a difference in luminance, typically the difference in reflected light levels between adjacent triangles. A CSF is usually expressed in units of cycles per degree, which in the case of the hybrid technique refers to the number of pixels that can be discerned at a particular distance from the POR.

Creation of a perceptually based intermediate mesh, defining points through which rays are cast, is achieved using a discretized approximation of an empirically derived contrast sensitivity function. This is created using a rearrangement of the contrast threshold function explored by Geisler and Perry [1998]. The derivation starts with the contrast threshold equation:

$$CT(f, e) = CT_0 \exp(\alpha f \frac{e + e_2}{e_2}), \quad (1)$$

where f is the spatial frequency in cycles per degree, e is the visual angle with respect to the POR (eccentricity), CT_0 is the minimum contrast threshold, α is the spatial frequency decay constant, and e_2 is the half-resolution eccentricity. Geisler and Perry determined values for $CT_0 = 1/64$, $\alpha = 0.106$, and $e_2 = 2.3$, empirically.

By setting the left hand side of (1) to the maximum contrast possible, i.e., 1 (unity), and rearranging terms we obtain: $e = (e_2/\alpha f) \ln(1/CT_0) - e_2$. The resulting contrast sensitivity function provides the eccentricity at which a given spatial frequency f no longer contributes to perception. In order to construct a spatial frequency map for a monitor's display it is necessary to compute e at coordinates (x, y). Rearranging terms again gives:

$$f(x, y) = \frac{e_2 \ln(1/CT_0)}{\alpha(e(x, y) + e_2)}. \quad (2)$$

The resulting spatial frequency map is shown in Figure 2. This map guides the sampling rate of the intermediate mesh by dictating the minimum cycles per degree at a particular point on the screen. The CSF-based ray mask generation process is as follows.

The first step is the creation of a 1-D array of pixel values representing rays to cast along the positive x-axis. The ratio between pixel extent and physical extent of the display is determined, allowing the eye/screen distance to be mapped into pixel space. With this information a set of equations representing the spatial frequency of casting a ray through every pixel (every two pixels, every three pixels, etc.) can be generated. These equations, in units of Pixels Per Degree (PPD), are obtained via: $\text{PPD}(n) = \tan(1.0)/n, n \in [1..17]$.

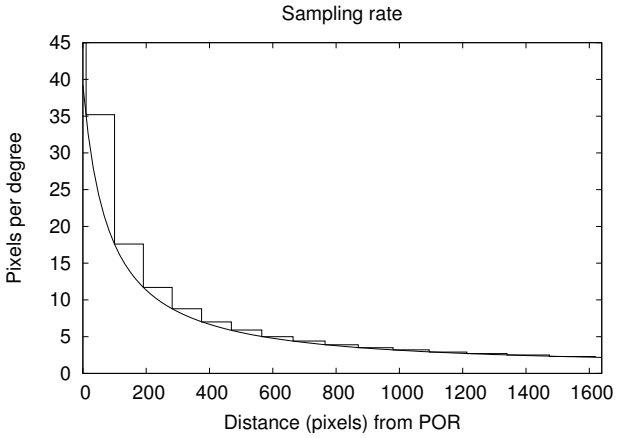

Figure 2: *Plot of minimum sampling rate derived from the CSF, with superimposed discretized version used to guide ray casting (at 53 cm viewing distance).*

The intersection points between this set of equations and (2) indicate pixel spacing transitions in the 1-D array. The 1-D array is populated by inserting every integer from 0 to the first intersection point, then every second integer from the first intersection to the second intersection, and so forth until a value in the 1-D array exceeds the maximum screen extent.

The second step is to generate the 2-D ray mask. Given the 1-D array of pixel locations (corresponding to rays to be cast), this is accomplished by rotating the pixel values by some angular displacement. The displacement is determined by calculating the separation of the outermost 2 entries in the 1-D array and converting them from a pixel separation into an angular separation (at the desired eye/screen distance). This angular displacement is used to rotate the points, whose positions are recorded in the 2-D ray mask, until a full 360° is attained. Any pixels within the distance indicated in the 1-D array's contiguous pixel region are filled in to account for discretization errors in the rotation process.

Finally, connectivity information is generated based on the ray mask. Each concentric circle of rays outside the contiguous region is joined with the succeeding circle of rays to form a quad strip. The combination of ray mask and connectivity information is rendered in place of the original scene geometry and is referred to as the intermediate mesh.

3.2 Ray Casting

Ray casting, a subclass of ray tracing, is a rendering technique that tests for intersections between a ray and primitives in the scene. In order to achieve efficient render times for any non-trivial scene it is necessary to reduce the number of ray/primitive intersection tests. The most obvious algorithmic approach is to use a ray casting acceleration structure, which places boundaries on the search space to be traversed in determining if an intersection has occurred. Modern ray casting acceleration structures fall into two bodies of solutions: Bounding Volume Hierarchies (BVHs) and spatial subdivision (e.g., see Purcell et al. [2002]).

BVHs partition the primitives in a scene using a bottom up approach. A primitive is placed within a bounding volume (typically a sphere or axis-aligned bounding box, for ease of intersection test-

ing). More primitives are placed within the bounding volume until a predetermined threshold is reached.

Spatial subdivision takes an alternate approach by seamlessly partitioning the space in a scene, rather than the primitives. Two common examples of this top-down approach are the octree and uniform grid. Octrees recursively organize space into octants, each of which contain either empty space or primitives to be tested. Uniform grid representations instead partition space into a 3-D array of identical rectangular prisms, which are traversed in much the same way as a line drawing algorithm traverses a 2-D array of pixels.

The decision to use a uniform grid in this experiment was based on several factors. Uniform grids tend to perform well for scenes with uniform primitive density, such as the search task scenes used in this experiment. In addition, the traversal algorithm for a uniform grid involves simple repetitive arithmetic, which allows a sequential representation in memory, as opposed to following links in a tree (cf. with Purcell et al.'s [2002] uniform grid rationale). The GPU used in the development of the hybrid technique was heavily penalized for texture look-ups, making link-based traversal, in which links can exist to widely separated memory locations, undesirable. Given that texture look-ups are inevitable, it was important to maintain the spatial locality of memory references to minimize texture cache misses. It was believed that storing a uniform grid by decomposing it into planes perpendicular to the major axis of the view vector (at successively greater distances from the eye location) would satisfy this requirement.

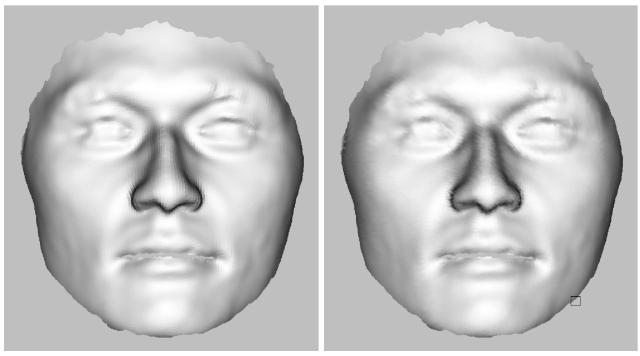

Figure 3: *Original rendition (left), non-isotropically degraded rendition (right) with silhouette edges maintained and POR superimposed at lower right.*

The process used to create a uniform grid acceleration structure for use with the hybrid technique begins with determining a bounding box for the entire mesh. This serves as the initial cell in the uniform grid. Mesh primitives are placed in the appropriate cell in the bounding box until any cell exceeds an empirically derived binning size (40 triangles per cell worked well for the meshes used in this experiment). Primitives that span more than one cell are placed in all intersected cells. Exceeding the binning size results in an increased level of subdivision for the bounding box. The process is repeated with the redefined uniform grid until all primitives are inserted without problem. Pertinent information, such as bounding box and cell dimensions, are recorded and stored with the uniform grid structure.

3.3 Hybrid Rendering

With both the CSF-based ray mask and uniform grid acceleration structure in place, using the hybrid technique is simple. Rays are cast from the eye point through each pixel indicated in the ray mask. The rays pass through cells in the uniform grid, where ray/primitive intersection tests are performed for all primitives present in the cell. If an intersection occurs the information is stored in the intermediate mesh and the ray's progress is halted. This process is repeated until all rays have been cast.

Following the initial scene sampling, the intermediate mesh is searched for quads with heterogeneous intersection data. Any quad with a mix of intersected and nonintersected corners indicates an external silhouette edge and is rendered by casting rays through all bounded pixels. All homogeneous quads are rendered using standard rasterization.

Implementation on a GPU required special consideration of some parts of the process. The ray mask was converted into a texture, with the RGB components storing the normalized XYZ direction of the ray. The uniform grid was stored in two separate textures. Each pixel in the first texture represented a cell in the grid and consisted of a 2-tuple defining the number of primitives present in the cell and the initial offset into a list of primitives, stored in the G and R components, respectively. The second texture stored the primitive lists, with each entry (a luminance only texture was requested) indicating an offset into the original geometry primitive list (which stored the actual XYZ coordinates of each triangle vertex). The original mesh geometry was also stored in RGB textures.

After loading the textures onto the GPU, a rectangle corresponding to the original bounding box is drawn on the screen. OpenGL rasterizes the rectangle into fragments, which are passed to a fragment shader. A simple comparison with the pixels in the ray mask texture (offset by the POR) culls non-ray fragments. Fragments surviving this stage progress through the uniform grid traversal shader. Each fragment "color" starts with all components initialized to 0.0; an intersection results in the alpha value being set to 1.0 and the RGB values set to the intersected triangle's normal.

The intersection data stored in the framebuffer is read into system memory and associated with quads in the intermediate mesh. All quads are then categorized by homogeneous and heterogeneous intersection types. The original OpenGL render pipeline state is restored and the list of homogeneous quads, with color information extracted from the copied framebuffer, is drawn. A full ray casting shader is loaded and the heterogeneous list is drawn, with each rasterized fragment resulting in a cast ray. The result is a three layer image: the POR rendered using the contiguous region of the ray mask, the homogeneous quads (internal mesh detail) rasterized on top of the initial ray casting results, and the heterogeneous quads (external silhouette edges) ray cast on top of both previous layers. The framebuffer is displayed and the process restarts.

3.4 GPU Implementation Issues

Shader programs used in this experiment were developed using the OpenGL Shading Language (GLSL). Although not all GPU specific capabilities were exposed, some of which may have been pertinent to accelerating the hybrid technique, GLSL proved satisfactory for the primary tasks of uniform grid traversal and ray/triangle intersection testing. There are, however, outstanding issues worth noting outside those discussed above.

Although implementation dependent, the most common upper bound on texture sizes in OpenGL is 4096×4096 pixels for any color representation (on modern hardware). While the resulting 16M floating point 4-tuples proved more than satisfactory to hold the mesh geometry, the uniform grid acceleration structure was more tightly constrained, consisting of an approximately $256 \times 256 \times 256$ element design. This was sufficient, but reducing the geometry binning size would have resulted in an overly large structure. One possible solution would be to span the uniform

grid acceleration structure over several textures; for example, eight 512×512×64 could be organized into a 512×512×512 element design.

Another issue affecting the uniform grid acceleration structure was the GPU's use of an 8-bit loop counter and maximum of 65536 instructions in a shader program. Again, this constrained the maximum number of elements traversed to 256, as nesting the loop to give greater depth ran the risk of exceeding the maximum number of instructions available in a shader.

The resultant refresh rate with silhouette preservation was informally measured at an average of 15 fps. The average refresh rates during full screen ray casting and GCD with no edges were 33 and 27 fps, respectively. Unfortunately, the current hardware's readback operation to system memory during edge detection conceals any potential rendering speedup due to gaze-contingent display. It seems likely that the suggestions in Section 7.1 and the rapidly advancing state of GPU technology may solve the aforementioned issues. Nevertheless, even with no clear rendering speed advantage, the silhouette-preserving GCD renders a functionally imperceptible degradation that affords a significant visual search performance benefit surprisingly even over full screen ray casting (see below).

4 Methodology

The objective of this experiment was to determine if the hybrid technique produced results comparable to existing gaze-contingent rendering techniques. It was hypothesized that varying the size of a high-resolution inset at the POR, while reducing peripheral detail in accordance with the CSF, would affect localization time in a visual search task.

In addition to testing for an inverse relationship between window size and search time, the flexibility of the hybrid technique to maintain external silhouette edges allowed a comparison between search times for scenes both with and without silhouette edges.

4.1 Equipment

A Tobii ET-1750 video-based binocular eye tracker was used for real-time gaze sampling. The Tobii samples at 50 Hz with an accuracy typically better than 0.3° over a $\pm 20^\circ$ horizontal and vertical range using the pupil/corneal reflection difference [Tobii Technology AB 2003] (in practice, measurement error ranges roughly $\pm$ 10 pixels). The eye tracker's $17''$ LCD monitor was set to 1280×1024 resolution. The eye tracking server ran on a dual 2.0 GHz AMD Opteron 246 PC (2 G RAM) running Windows XP. Although the Tobii allows limited head movement (30×15×20 cm volume), a chinrest was used to maintain constant distance (53 cm) from the monitor.

The client display application ran on a 2.2 GHz AMD Opteron 148 Sun Ultra 20 running the CentOS operating system. The client/server PCs were connected via the departmental 1 Gb Ethernet (both connected to a switch on the same subnet). A keyboard attached to the client system provided user input. The physical setup is shown in Figure 4 (viewer is running a demonstration program).

The client PC was equipped with an Nvidia 8800GTX GPU with 768Mb of texture memory and 128 stream processors operating at 1.35 GHz. All ray casting occurred on-GPU, however, construction of the intermediate mesh necessitated CPU involvement.

4.2 Experimental Design & Procedure

The independent variables used in this 2×5 repeated measures experiment were high-resolution inset size and the presence or absence of maintained object edges. Inset sizes spanned 2°, 5°, 10°, 15°, and 20° visual angle. For each inset size, object edges were either maintained or discarded, resulting in 10 combinations. Addition of a control (full screen ray casting) resulted in a total of 11 scene types to be tested.

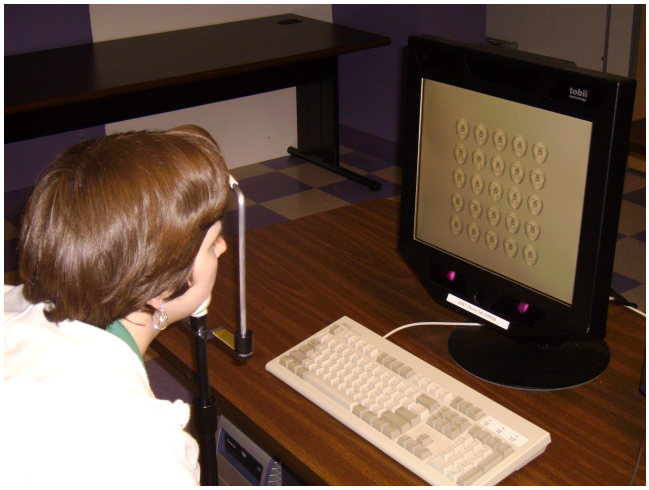

Figure 4: *Example of equipment setup.*

The dependent variable in this experiment was the visual search time required to localize a target object. The user was simply asked to find a replica of the centrally located target object in a field of distractors of similar appearance. Users were first familiarized with an isolated image of the target object, then, when ready (no time limit imposed), the screen changed to display a 5×5 object search field, with the target located at center and simultaneously at a random position in the grid of distractors (see below). Search time started when the field of distractors was displayed and ended when the subject pressed the Space key.

The experiment utilized a within-subjects design, with each participant exposed to the aforementioned 11 scene types. A Latin square was used to control order effects. Each subject performed the 11 search tasks 4 times, resulting in 44 total trials.

4.3 Human Subjects

There were 11 participants in the experiment, 4 female and 7 male. Subject ages ranged from 25 to 34, all with self-reported normal or corrected to normal vision. Although contact lenses were permitted, potential subjects wearing glasses were excluded from the experiment, as were others who failed the screening process (a total of 2) due to abnormally inconsistent eye tracking data.

4.4 Stimulus

A 9 point calibration was used at the beginning of a session, with a calibration accuracy confirmation scene displayed immediately thereafter. The calibration accuracy confirmation scene was also displayed at the mid-point of the experiment to assure that accuracy was within tolerances.

Four topologically consistent meshes displaying different facial expressions were used as targets. The original silhouette edges were consistent across all four meshes. Each target subtended approximately 3.5 degrees of visual angle at the screen distance used in the experiment. The entire scene subtended approximately 20 degrees of visual angle.

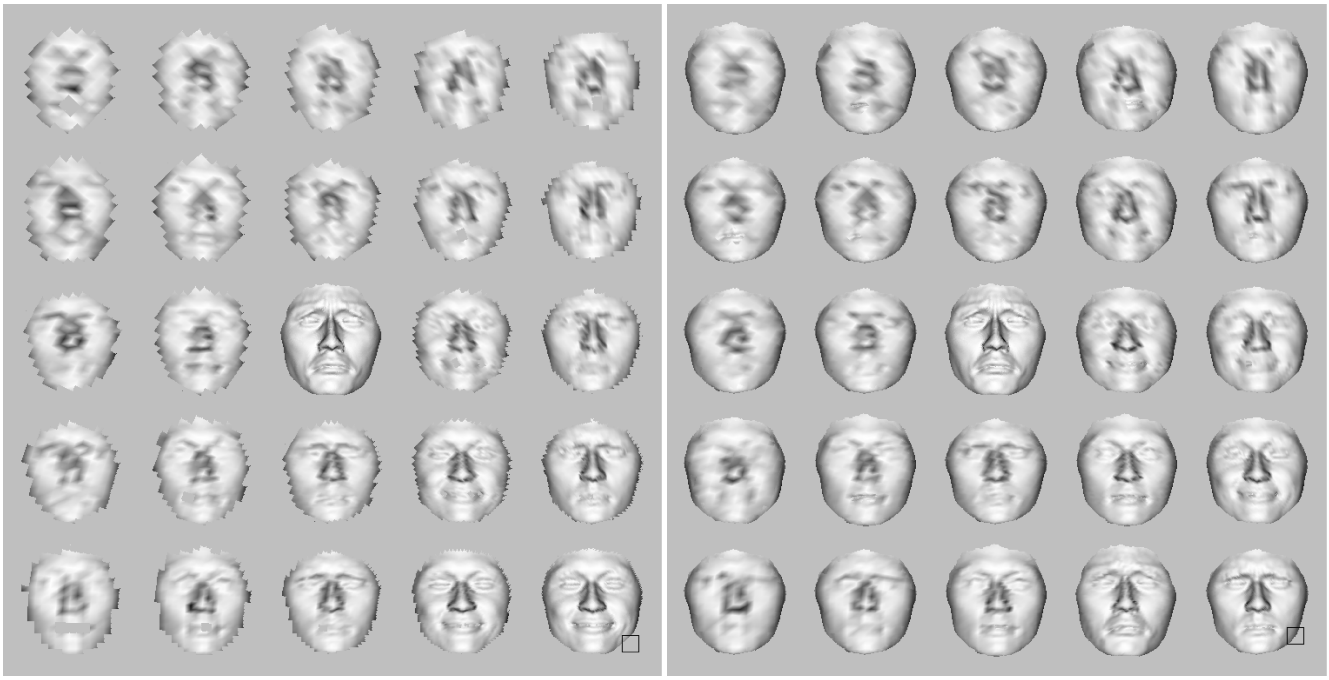

Figure 5: *Sample search task (left), sample search task with silhouette edges preserved (right), each with box in lower right indicated POR.*

The scenes used for the visual search task consisted of a 5×5 grid of the meshes described above. The scenes were constructed by first selecting a target from the set of four meshes. Each mesh was selected at random from a pre-seeded pool, assuring that although the presentation order of each mesh was random, each mesh assumed the role of target the same number of times. The target mesh was then placed both in the center of the scene and at a random location in the grid. The remaining positions in the 5×5 grid were then filled at random with duplicates of the three non-target meshes.

4.5 Presentation

A scripted verbal explanation of the display program commands and execution flow was given to familiarize subjects with the demonstration program. The chair, chinrest, and keyboard were then adjusted for user comfort. At this point a demonstration program was run to reinforce the scripted explanation regarding the commands and program flow to be used in the experiment. The subjects were informed that the demonstration program functioned in exactly the same manner as the experimental program, with only the control group being displayed. The demonstration program was also used to introduce the meshes to be used in the experiment. Subjects were informed that time was not a factor in the demonstration, and were encouraged to examine the different facial expressions in detail. Upon completion of the demonstration program the participants were permitted to ask questions and stretch.

Subjects started the experimental program when ready. A 9 point calibration was followed by a scene designed to allow the experimenter to judge the quality of the calibration. Recalibration was performed if the calibration was deemed to lack accuracy or precision. Following calibration, the target was displayed in the center of the screen. The subject was given as much time as needed to examine the target. The subject then triggered the search task, which displayed the 5×5 grid of distractors. Upon localizing the target within the grid, the participant pressed the Space key to finish the search task and load the next target. A break to check the calibration or stretch was allowed after 6 search tasks were completed. Upon completion of all 11 search tasks, users were encouraged to stand and stretch. The experimental program was run 4 consecutive times for each subject.

5 Results

One-way ANOVA indicates the effect of silhouette edges on time to completion is significant ($F(2,481) = 7.301$, $p < 0.01$). Pairwise t-tests with pooled SD suggest that visual search performance differs significantly between trials with edges maintained and trials with degraded (no) edges ($p < 0.01$; see Figure 6 left).

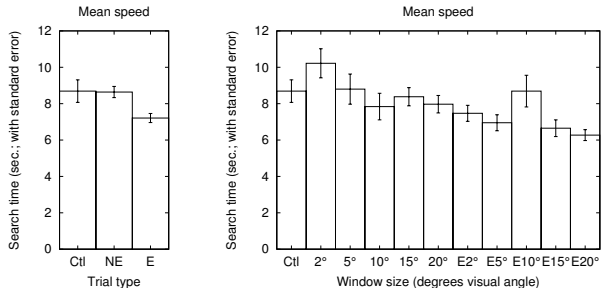

Figure 6: *Mean search times (pooled, left; per window size condition, right; 'E' = silhouette edge preservation, 'NE' = no edges).*

One-way ANOVA indicates the effect of window size on time to completion is significant ($F(10,473) = 3.381$, $p < 0.01$). Pair-wise t-tests with pooled SD indicate significantly slower search times with a 2° window with degraded silhouette edges than with windows (with silhouette edges preserved) of 5° ($p < 0.05$) and 15° and 20° ($p < 0.01$; see Figure 6 right). A statistically significant difference was also expected between the edge-preserved window of 10° but none was observed.

6 Discussion

As expected, search time decreased as window size increased and was lower still when silhouette edges were maintained. Search time was significantly prolonged with a 2° window with no silhouette edge information. Results suggest that performance is degraded with a very small foveal window (2°) when edge information is not preserved.

Given that the scene sampling rate was driven by the CSF, it is not surprising that the results of this experiment are similar to other gaze-contingent rendering experiments. The trend shown in Figure 6, an inverse relationship between high-resolution inset and visual search time, is consistent with previous image-based GCD results, e.g., those of Watson et al. [2004], who evaluated a visual search task over face images with a $30^\circ \times 30^\circ$ high-resolution inset. Peripheral degradation was uniform within a scene (as opposed to the continuously varied peripheral degradation used in the hybrid technique), but varied between trials.

Our results are also similar to those of Parkhurst and Niebur [2004], obtained for their model-based visual search task. Their results are based on peripheral degradation generated through edge collapse, guided by normalized error in the collapse process. It should be noted that the visual search task utilized objects that varied widely in topology and profile.

In our case, a typical search would start at the center of the display and sweep out concentrically in search of the target. For example, as shown in Figure 7, targets are fixated in counter clockwise order until the target is located at upper left, when it is visually verified against the reference mesh at center (note the multiple refixations).[1] Not surprisingly, fixations tend to cover the mouth region, a particularly distinguishing feature of the mesh.

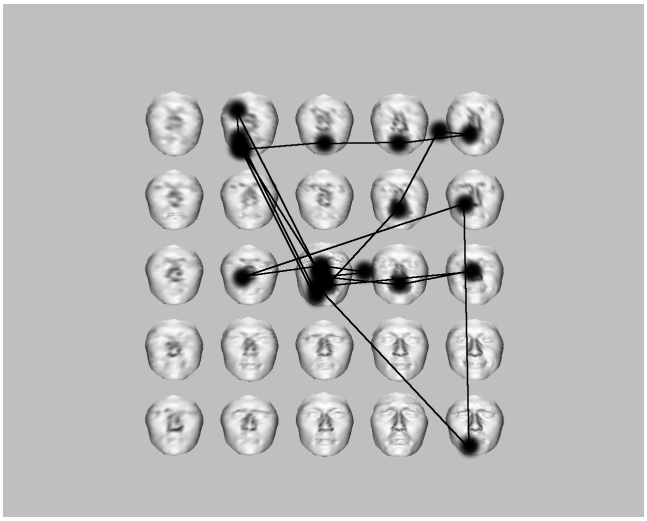

Figure 7: *Example scanpath exhibited during visual search, in this case starting at center and sweeping out concentrically in a counter clockwise direction until target is located at upper left (when it is visually verified against the reference mesh at screen center).*

Two somewhat surprising artifacts appear in our data. First, the relatively long mean search time for the 10° window with preserved edges (E10) appears to be anomalous. A Latin square design was employed to reduce ordering effects, making it unlikely that a learning effect caused the increased search time. It should be noted that the standard error for E10 was the highest of any mean search time.

Second, the relatively poor performance of the Control group was unexpected. In the absence of any peripheral degradation, subjects were expected to localize targets rapidly. Anecdotal evidence, acquired through informal exit interviews, indicates that the Control scenes were "overwhelming" and provided "too many possibilities". It appears that the lack of overt visual cues, coupled with the rich informational environment, precluded rapid localization of peripheral targets. There is some support in the literature for this interpretation. Specifically, Cave and Bichot [1999] suggest that for discrimination tasks, peripheral objects may increase reaction time if they distract the viewer during discrimination. Furthermore, the authors point out that the importance of attention in suppressing competing information from distractors is supported by neurophysiological studies showing that attentional modulation of neural responses is greatest when target and distractor both fall in the receptive field of a neuron, thus competing for representation by that neuron. By masking peripheral distractors, a GCD may be beneficial to visual search where discrimination is a significant task component. The reader should be cautioned that this interpretation, although apparently in agreement with our observations, is rather tenuous. More controlled experimentation is required to substantiate this claim.

7 Conclusions

A non-isotropic gaze-contingent rendering technique was presented utilizing aspects of both imaged-based and model-based rendering. The experimental results of the hybrid technique conform to prior results obtained using both image- and model-based techniques. It would be instructive to duplicate the experiment using image- and model-based degradation in order to directly compare the search times, but until the most recent generation of GPU, the geometry manipulation required of model-based techniques forced the algorithms to run on the CPU.

7.1 Future Work

There are several areas of potential improvement that may increase the efficiency of the hybrid technique. The decision to use a uniform grid was partially based on the architectural restrictions specific to the GPU used during development (an older graphics card). Use of the much more capable 8800GTX alleviates some of those concerns and should allow the use of more computationally efficient acceleration structures, such as a bounding volume hierarchy.

Another GPU-based improvement is to exploit the on-chip geometry instantiation made available on the 8800GTX. Currently, the entire frame buffer is copied back into system memory after rays are cast and accessed to generate the intermediate mesh, which is then sent back to the GPU to be rendered. It may be possible to create the intermediate mesh directly on the video card through the use of geometry shaders, simultaneously reducing traffic on the system bus and further exploiting the parallel nature of the GPU. This would also allow the possibility of adaptive subdivision of the intermediate mesh, reducing the number of rays cast to maintain silhouette edges.

An alternative to more efficient exploitation of hardware would be greater utilization of the HVS. The CSF function used was intentionally forced into its most conservative form. It is possible to increase peripheral degradation either by setting (1) to a non-maximal value of contrast, or by taking into account the kinesthetic portion of the CSF (cf. Daly et al. [2001]).

[1] Fixation are detected via velocity-based analysis where velocity $< 15^\circ$/s denotes a fixation; the centroid of all congruent fixations determines the mean fixation coordinates.

Dynamic generation of CSF-based ray masks would also benefit the hybrid technique. The use of a chinrest during the experiment was necessary to keep the subject's eyes at the proper distance from the screen for the CSF function used. If the eye/screen separation could be determined on-the-fly, distance-correct ray masks could offer improved performance by sampling a scene less frequently.

Acknowledgments

Thanks to Robert Sumner and Jovan Popović of the Computer Graphics Group at MIT for providing the mesh data used in this experiment.

Thanks to Jacob Richards at Blue Sky Studios for Maya services "rendered".

References

BERGSTRÖM, P. 2003. *Eye-movement Controlled Image Coding*. PhD thesis, Linköping University, Linköping, Sweden.

CAVE, K. R., AND BICHOT, NARCISSE, P. 1999. Visuospatial Attention: Beyond a Spotlight Model. *Psychonomic Bulletin & Review 6*, 2, 204–223.

CLARKE, J. H. 1976. Hierarchical Geometric Models for Visible Surface Algorithms. *Communications of the ACM 19*, 10 (October), 547–554.

DALY, S., MATTHEWS, K., AND RIBAS-CORBERA, J. 2001. As Plain as the Noise on Your Face: Adaptive Video Compression Using Face Detection and Visual Eccentricity Models. *Journal of Electronic Imaging 10*, 1, 30–46.

DANFORTH, R., DUCHOWSKI, A., GEIST, R., AND MCALILEY, E. 2000. A Platform for Gaze-Contingent Virtual Environments. In *Smart Graphics (Papers from the 2000 AAAI Spring Symposium, Technical Report SS-00-04)*, AAAI, 66–70.

DUCHOWSKI, A. T., AND ÇÖLTEKIN, A. 2007. Foveated Gaze-Contingent Displays for Peripheral LOD Management, 3D Visualization, and Stereo Imaging. *Transactions on Multimedia Computing, Communications and Applications 3*, 4 (November). Accepted for Publication.

DUCHOWSKI, A. T. 2007. *Eye Tracking Methodology: Theory & Practice*, 2nd ed. Springer-Verlag, Inc., London, UK.

GEISLER, W. S., AND PERRY, J. S. 1998. Real-time foveated multiresolution system for low-bandwidth video communication. In *Human Vision and Electronic Imaging*, SPIE.

GEISLER, W. S., PERRY, J. S., AND NAJEMNIK, J. 2006. Visual Search: The Role of Peripheral Information Measured Using Gaze-Contingent Displays. *Journal of Vision 6*, 9, 858–873.

HOPPE, H. 1997. View-Dependent Refinement of Progressive Meshes. In *Computer Graphics (SIGGRAPH '97)*, ACM.

LEVOY, M., AND WHITAKER, R. 1990. Gaze-Directed Volume Rendering. In *Computer Graphics (SIGGRAPH '90)*, ACM, 217–223.

LINDSTROM, P., KOLLER, D., RIBARSKY, W., HODGES, L. F., FAUST, N., AND TURNER, G. A. 1996. Real-Time, Continuous Level of Detail Rendering of Height Fields. In *Computer Graphics (SIGGRAPH '96)*, ACM, 109–118.

LUEBKE, D., AND ERIKSON, C. 1997. View-Dependent Simplification Of Arbitrary Polygonal Environments. In *Computer Graphics (SIGGRAPH '97)*, ACM.

LUEBKE, D., AND HALLEN, B. 2001. Perceptually Driven Simplification for Interactive Rendering. In *Proceedings of the 2001 Eurographics Workshop on Rendering (Rendering Techniques)*, S. Gortler and K. Myszkowski, Eds., Eurographics.

LUEBKE, D., HALLEN, B., NEWFIELD, D., AND WATSON, B. 2000. Perceptually Driven Simplification Using Gaze-Directed Rendering. Tech. Rep. CS-2000-04, University of Virginia.

MACCRACKEN, R., AND JOY, K. 1996. Free-From Deformations With Lattices of Arbitrary Topology. In *Computer Graphics (SIGGRAPH '96)*, ACM, 181–188.

MURPHY, H., AND DUCHOWSKI, A. T. 2001. Gaze-Contingent Level Of Detail. In *EuroGraphics*, EuroGraphics.

OHSHIMA, T., YAMAMOTO, H., AND TAMURA, H. 1996. Gaze-Directed Adaptive Rendering for Interacting with Virtual Space. In *Proceedings of VRAIS'96*, IEEE, 103–110.

PARKHURST, D. J., AND NIEBUR, E. 2002. Variable Resolution Displays: A Theoretical, Practical, and Behavioral Evaluation. *Human Factors 44*, 4, 611–629.

PARKHURST, D. J., AND NIEBUR, E. 2004. A Feasibility Test for Perceptually Adaptive Level of Detail Rendering on Desktop Systems. In *Applied Perception, Graphics & Visualization (APGV) Symposium*, ACM, 49–56.

PURCELL, T. J., BUCK, I., MARK, W. R., AND HANRAHAN, P. 2002. Ray Tracing on Programmable Graphics Hardware. *ACM Transactions on Graphics 21*, 3 (July), 703–712. ISSN 0730-0301 (Proceedings of ACM SIGGRAPH 2002).

REDDY, M. 1998. Specification and Evaluation of Level of Detail Selection Criteria. *Virtual Reality: Research, Development and Application 3*, 2, 132–143.

REINGOLD, E. M., LOSCHKY, L. C., MCCONKIE, G. W., AND STAMPE, D. M. 2002. Gaze-Contingent Multi-Resolutional Displays: An Integrative Review. *Human Factors 45*, 2, 307–328.

SCHMALSTIEG, D., AND SCHAUFLER, G. 1997. Smooth Levels of Detail. In *Proceedings of VRAIS'97*, IEEE, 12–19.

TOBII TECHNOLOGY AB, 2003. Tobii ET-17 Eye-tracker Product Description (v1.1). URL: <http://www.tobii.se/> (last accessed January 2007).

WATSON, B., WALKER, N., HODGES, L. F., AND WORDEN, A. 1997. Managing Level of Detail through Peripheral Degradation: Effects on Search Performance with a Head-Mounted Display. *ACM Transactions on Computer-Human Interaction 4*, 4 (December), 323–346.

WATSON, B., WALKER, N., AND HODGES, L. F. 2004. Supra-Threshold Control of Peripheral LOD. *Transaction on Graphics (SIGGRAPH'04 Proceedings) 23*, 3, 750–759.

ZORIN, D., AND SCHRÖDER, P. 2000. *Course 23: Subdivision for Modeling and Animation*. ACM SIGGRAPH, New York, NY. URL: <http://www.mrl.nyu.edu/dzorin/sig00course/> (last accessed 12/30/00).

ZORIN, D., SCHRÖDER, P., AND SWELDENS, W. 1997. Interactive Multiresolution Mesh Editing. In *Computer Graphics (SIGGRAPH '97)*, ACM.